CS 545 Machine Learning

Homework 2

By: Mohak Patel

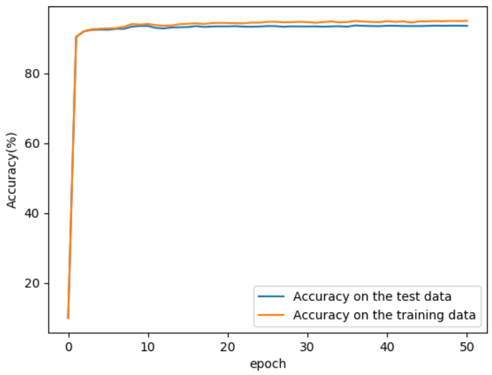
Description:

This experiment is about implementing the multilayer perceptron learning algorithm. In this algorithm, training and testing are done on MNIST dataset. The MNIST dataset contains 60,000 training data and 10,000 test data of handwritten digits. Perceptron will learn to recognize handwritten digit. There is a 10 perceptron, as a group it will classify the hand written digit. Each perceptron has 785 inputs and one output. Each perceptron's target is one of the 10 digits (0-9). Input is scaled down between 0 and 1 by dividing the input value with 255. Perceptron's weights value is a random value between -0.05 and 0.05. Perceptron will train the model on three different number of hidden units, momentum and number of training examples. The training period is for 50 epoch. After each epoch, the accuracy of training and testing is compared and plot the graph for epoch vs accuracy. It is shown below.

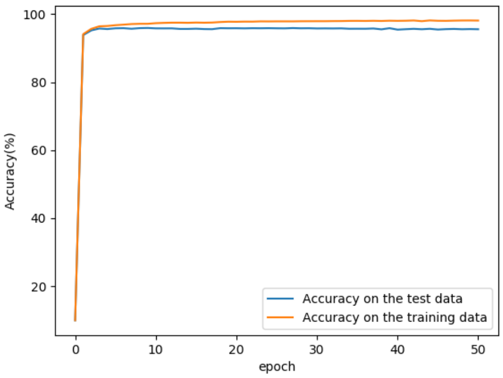
Experiment [1]: Vary the number of hidden units

In this experiment, we observe the test accuracy with a different number of hidden units. For this experiment learning rate=0.1 and momentum=0.9. The results shown below is for 20, 50 and 100 hidden units in the hidden layer.

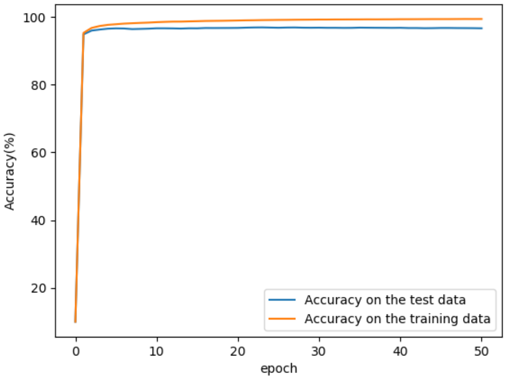
Results:



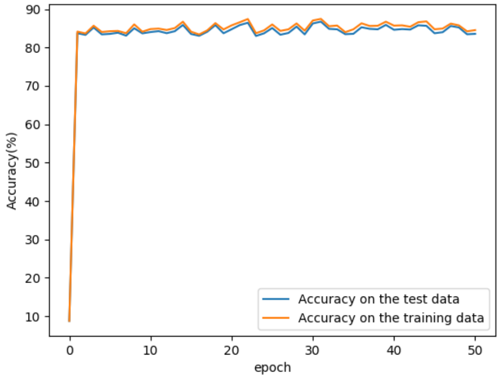
(a) Hidden units: 20



(b) Hidden units: 50



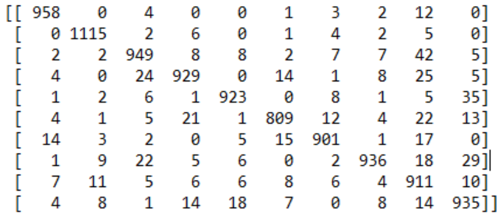
(C) Hidden units:100



(D) No hidden layer

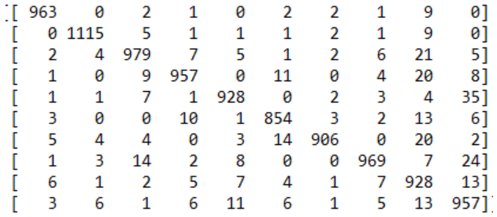
We can observe from the first three graphs that as we increased the hidden units in the hidden layer, test accuracy also increased. For the hidden units 20, 50 and 100 the maximum test accuracy is 93.66, 95.4 and 97.1, respectively. From the graphs, we can observe that we required less number of epoch for training to converge, as we increased the number of hidden units in the hidden layer. For hidden units 20, 50 and 100, it takes the 12 to 15 epochs, 8 to 10 epochs and 5 epochs for training to converge, respectively. We can clearly see in the 2nd and 3rd graph that both graphs has overfitting. It is because of the more number of hidden units and more epochs. Due to this two reason the neural network can easily remember the training data and it cannot predict the unseen data. As we can observe from the 4th graph that it has oscillation because there is no hidden layer and momentum in the single layer perceptron. And we can observe that there is no convergence in single layer perceptron.

Confusion Matrix:



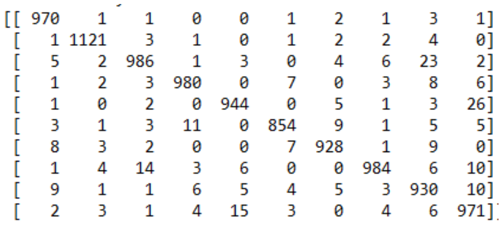
```
[[ 958    0    4    0    0    1    3    2   12    0]
 [   0 1115    2    6    0    1    4    2    5    0]
 [   2    2  949    8    8    2    7    7   42    5]
 [   4    0   24  929    0   14    1    8   25    5]
 [   1    2    6    1  923    0    8    1    5   35]
 [   4    1    5   21    1  809   12    4   22   13]
 [  14    3    2    0    5   15  901    1   17    0]
 [   1    9   22    5    6    0    2  936   18   29]
 [   7   11    5    6    6    8    6    4  911   10]
 [   4    8    1   14   18    7    0    8   14  935]]
```

Hidden units:20



```
[ 963    0    2    1    0    2    2    1    9    0]
 [   0 1115    5    1    1    1    2    1    9    0]
 [   2    4  979    7    5    1    2    6   21    5]
 [   1    0    9  957    0   11    0    4   20    8]
 [   1    1    7    1  928    0    2    3    4   35]
 [   3    0    0   10    1  854    3    2   13    6]
 [   5    4    4    0    3   14  906    0   20    2]
 [   1    3   14    2    8    0    0  969    7   24]
 [   6    1    2    5    7    4    1    7  928   13]
 [   3    6    1    6   11    6    1    5   13  957]]
```

Hidden units:50



```
[[ 970    1    1    0    0    1    2    1    3    1]
 [   1 1121    3    1    0    1    2    2    4    0]
 [   5    2  986    1    3    0    4    6   23    2]
 [   1    2    3  980    0    7    0    3    8    6]
 [   1    0    2    0  944    0    5    1    3   26]
 [   3    1    3   11    0  854    9    1    5    5]
 [   8    3    2    0    0    7  928    1    9    0]
 [   1    4   14    3    6    0    0  984    6   10]
 [   9    1    1    6    5    4    5    3  930   10]
 [   2    3    1    4   15    3    0    4    6  971]]
```

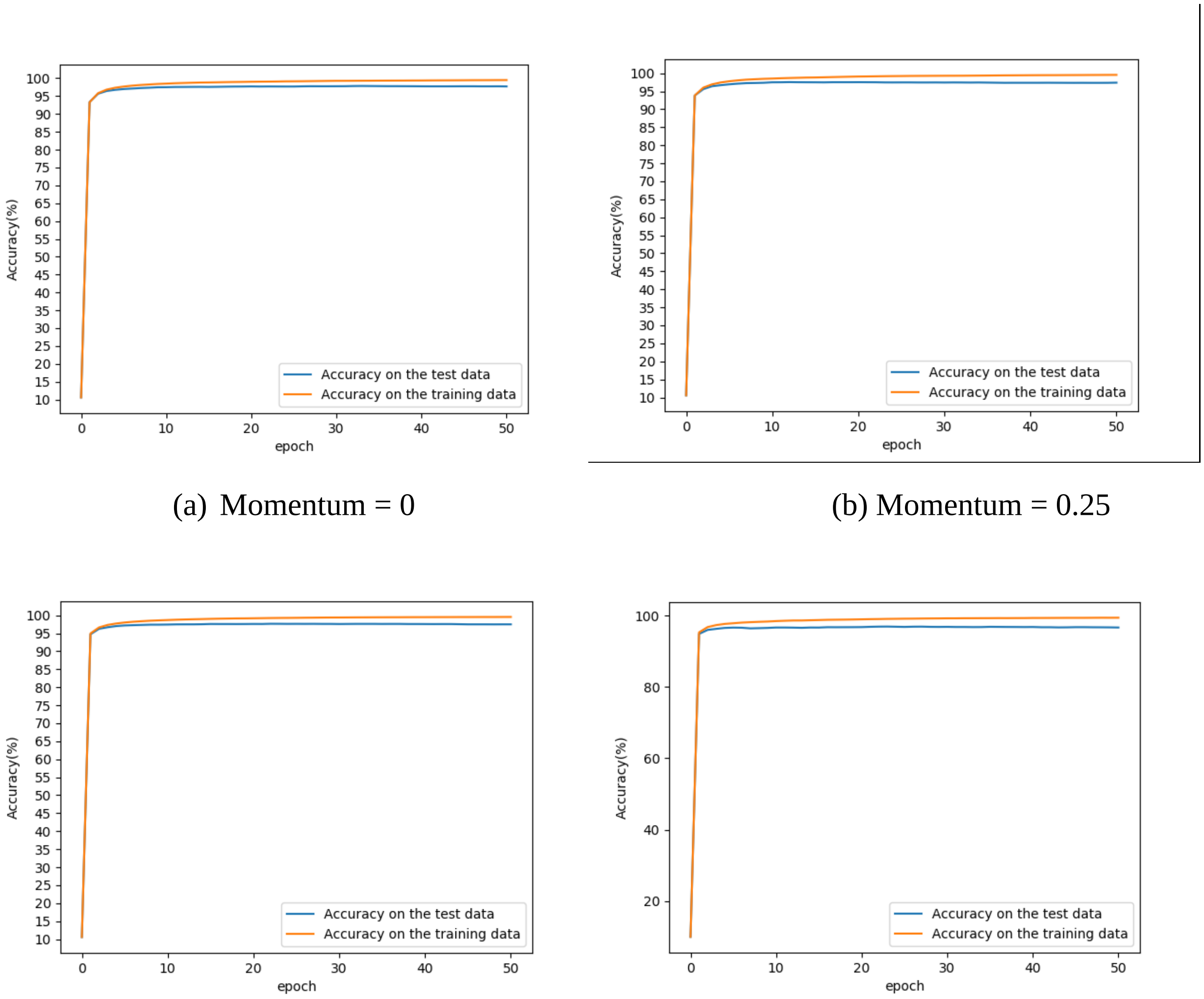
Hidden units:100

Here, each row represents the actual digit (0-9 respectively) and each column represents the predicted digit (0-9 respectively). From the confusion matrix, we can observe that most accurately predicted digit is 1 and least accurately predicted digit or most confused digit is 2 with 8.

Experiment [2]: Vary the momentum value.

In this experiment, the results show the changes in the accuracy of test data with a different value of momentum. Here the momentum is 0, 0.25 and 0.5. Learning rate is 0.1. We fix the number of hidden units (100) in the hidden layer.

Results:



(a) Momentum = 0

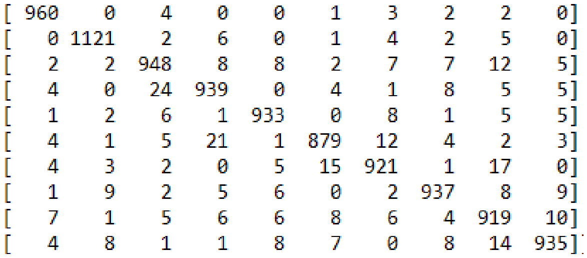
(b) Momentum = 0.25

(C) Momentum=0.5

(d) Momentum=0.9

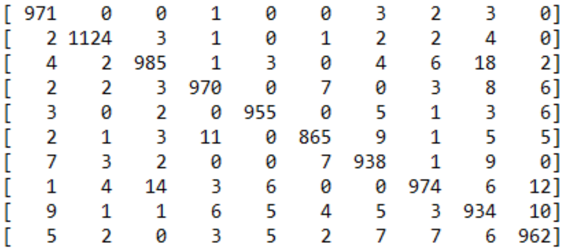
We can observe from the graph that as momentum increased, the test accuracy reaches the maximum in less number of the epoch. In the 1st graph we can see that the test accuracy gradually increased for epoch 3 to 6 but in the 4th graph for momentum 0.9, the test accuracy take less number of the epoch (2 to 3) to get maximum. As momentum increased, it takes less number of epochs for training to converge. We can observe from the graphs that high momentum helps the speed of convergence of the system. We can clearly see in the 3rd and 4th graph that both graphs has overfitting. It is because of the more number of hidden units and more epochs with high momentum. Due to this two reason, the neural network can easily remember the training data and it cannot predict the unseen data.

Confusion Matrix:



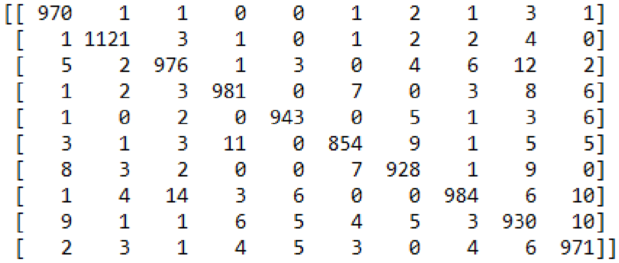
```
[ 960    0    4    0    0    1    3    2    2    0]
[   0 1121    2    6    0    1    4    2    5    0]
[   2    2  948    8    8    2    7    7   12    5]
[   4    0   24  939    0    4    1    8    5    5]
[   1    2    6    1  933    0    8    1    5    5]
[   4    1    5   21    1  879   12    4    2    3]
[   4    3    2    0    5   15  921    1   17    0]
[   1    9    2    5    6    0    2  937    8    9]
[   7    1    5    6    6    8    6    4  919   10]
[   4    8    1    1    8    7    0    8   14  935]
```

(a) Momentum = 0



```
[ 971    0    0    1    0    0    3    2    3    0]
[   2 1124    3    1    0    1    2    2    4    0]
[   4    2  985    1    3    0    4    6   18    2]
[   2    2    3  970    0    7    0    3    8    6]
[   3    0    2    0  955    0    5    1    3    6]
[   2    1    3   11    0  865    9    1    5    5]
[   7    3    2    0    0    7  938    1    9    0]
[   1    4   14    3    6    0    0  974    6   12]
[   9    1    1    6    5    4    5    3  934   10]
[   5    2    0    3    5    2    7    7    6  962]
```

(b) Momentum = 0.25



```
[[ 970    1    1    0    0    1    2    1    3    1]
 [   1 1121    3    1    0    1    2    2    4    0]
 [   5    2  976    1    3    0    4    6   12    2]
 [   1    2    3  981    0    7    0    3    8    6]
 [   1    0    2    0  943    0    5    1    3    6]
 [   3    1    3   11    0  854    9    1    5    5]
 [   8    3    2    0    0    7  928    1    9    0]
 [   1    4   14    3    6    0    0  984    6   10]
 [   9    1    1    6    5    4    5    3  930   10]
 [   2    3    1    4    5    3    0    4    6  971]]
```

(C) Momentum = 0.5

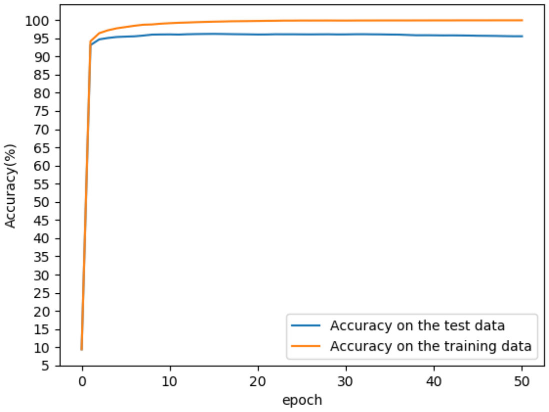
Here, each row represents the actual digit (0-9 respectively) and each column represents the predicted digit (0-9 respectively). From the confusion matrix, we can observe that most accurately predicted digit is 1 and least accurately predicted digit or most confused digit is 2 with 8.

<u>Experiment [3]:</u> Vary the number of training examples.

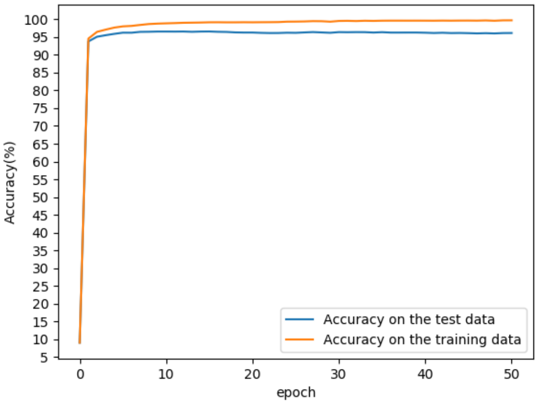
In this experiment, we fix the number of hidden units 100, momentum 0.9 and learning rate 0.1. We observe the results with a different number of the training example. Results show accuracy for the one quarter (15,000) and one half (30,000) of the training data. Training data is equally balanced among the different class of digit.

<u>Results:</u>

For the training examples 15000, the maximum accuracy of the test data is 95.34%. For the training examples 30000, the maximum accuracy of the test data is 96.84%. So the final test accuracy increased by approx. 1%, as we train the model with almost double training examples. And even from the above two experiments, for 60000 training example, we get maximum 97.12% accuracy on test data. Thus, for this experiment, as we increased the training examples from 150000 to 60000, final test accuracy also increased. Adding more examples adds diversity and it decreases the generalization error.



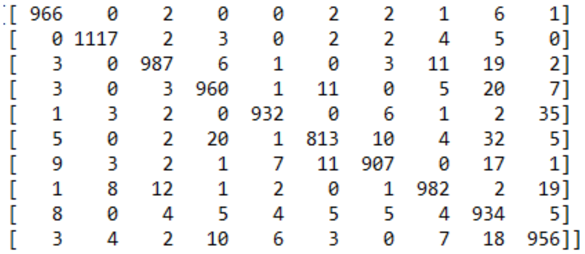
(a) Training data: 15,000



(b) Training data: 30,000

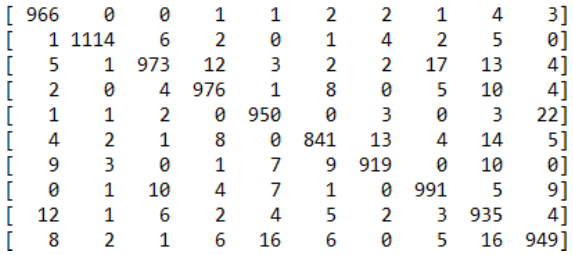
As we increased the training example, it takes less number of epochs for training to converge. We can observe that in the graphs. For 15000 data, it takes approx. 10 epochs for training to converge, where for 30000 data, it takes approx. 5 to 7 epochs for training to converge. We can observe from the graph that as we increased the training examples, overfitting also increased. It is because of the more number of hidden units and more epochs with high momentum. Due to this two reason, the neural network can easily remember the training data and it cannot predict the unseen data.

Confusion Matrix:



```
[ 966    0    2    0    0    2    2    1    6    1]
[   0 1117    2    3    0    2    2    4    5    0]
[   3    0  987    6    1    0    3   11   19    2]
[   3    0    3  960    1   11    0    5   20    7]
[   1    3    2    0  932    0    6    1    2   35]
[   5    0    2   20    1  813   10    4   32    5]
[   9    3    2    1    7   11  907    0   17    1]
[   1    8   12    1    2    0    1  982    2   19]
[   8    0    4    5    4    5    5    4  934    5]
[   3    4    2   10    6    3    0    7   18  956]]
```

(a) Training data: 15000



```
[ 966    0    0    1    1    2    2    1    4    3]
[   1 1114    6    2    0    1    4    2    5    0]
[   5    1  973   12    3    2    2   17   13    4]
[   2    0    4  976    1    8    0    5   10    4]
[   1    1    2    0  950    0    3    0    3   22]
[   4    2    1    8    0  841   13    4   14    5]
[   9    3    0    1    7    9  919    0   10    0]
[   0    1   10    4    7    1    0  991    5    9]
[  12    1    6    2    4    5    2    3  935    4]
[   8    2    1    6   16    6    0    5   16  949]
```

(b) Training data: 30000

Here, each row represents the actual digit (0-9 respectively) and each column represents the predicted digit (0-9 respectively). From the confusion matrix, we can observe that most accurately predicted digit is 1 and least accurately predicted digit or most confused digit is 4 with 9. As we can observe from the first confusion matrix, it has more number of false prediction compare to second confusion matrix.